Attorney Docket No.: 06502.0577

UNITED STATES PATENT APPLICATION

FOR

DYNAMIC PROVISIONING OF IDENTIFICATION SERVICES IN A DISTRIBUTED SYSTEM

BY

JAMES CLARKE

AND

SEAN CLARK

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

DYNAMIC PROVISIONING OF IDENTIFICATION SERVICES IN A DISTRIBUTED SYSTEM

DESCRIPTION OF THE INVENTION

Related Applications

[001] This application is a continuation-in-part of U.S. Patent Application No. 09/947,528 for Dynamic Provisioning of Service Components in a Distributed System, filed September 7, 2001, which is incorporated herein by reference. This application is related to U.S. Patent Application No. 09/947,549 for Distributed Metric Discovery and Collection in a Distributed System, filed September 7, 2001, and U.S. Patent Application No. 10/390,895 for Systems and Methods for Providing Dynamic Quality of Service for a Distributed System, filed March 19, 2003, which are relied upon and incorporated by reference.

Field of the Invention

[002] This invention relates to techniques for providing identification services in a distributed system and, more particularly, to methods and systems for provisioning services to process identification data received from a device.

Background of the Invention

[003] Distributed systems today enable a device connected to a communications network to take advantage of services available on other devices located throughout the network. Each device in a distributed system may have its own internal data types, its own address alignment rules, and its own operating system. To enable such heterogeneous devices to communicate and interact successfully,

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

developers of distributed systems can employ a remote procedure call (RPC) communication mechanism.

[004] RPC mechanisms provide communication between processes (e.g., programs, applets, etc.) running on the same machine or different machines. In a simple case, one process, i.e., a client, sends a message to another process, i.e., a server. The server processes the message and, in some cases, returns a response to the client. In many systems, the client and server do not have to be synchronized. That is, the client may transmit the message and then begin a new activity, or the server may buffer the incoming message until the server is ready to process the message.

[005] The Java™ programming language is an object-oriented programming language frequently used to implement such a distributed system. A program written in the Java programming language is compiled into a platform-independent format, using a bytecode instruction set, which can be executed on any platform supporting the Java virtual machine (JVM). The JVM may be implemented on any type of platform, greatly increasing the ease with which heterogeneous machines can be federated into a distributed system.

[006] The Jini™ architecture has been developed using the Java programming language to enable devices in a distributed system to share services using remote method invocation (RMI). Traditional Jini systems use RMI to enable a client device to request and receive a service provided by a server device on a remote machine. While conventional Jini systems provide a basic architecture for providing services in a distributed system, they do not provide tools specifically directed to providing complex services. Current systems do not address provisioning a service, such as application

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

software, to make it available to the distributed system in the first place. Furthermore, conventional systems do consider the requires of a specific service before provisioning the service to make it available in the distributed system.

[007] For example, one such distributed system may include reader devices using Radio Frequency Identification (RFID) technology, services that receive and process identification data from the reader devices, and applications that store and manage the processed data. Although RFID technology is well known, conventional RFID systems do not operate satisfactorily in a distributed system, where network changes, such as the failure of a computer resource or the introduction of a new resource, are common. It is therefore desirable to provide RFID systems that operates in a distributed, robust, reliable, scalable manner.

SUMMARY OF THE INVENTION

[008] Methods and systems consistent with the present invention provide identification services over a distributed network. Systems and methods provide the tools to receive data from reader devices, process the received data, and provide the processed data to users. Enhanced event handling and dynamic service provisioning enable robust and flexible deployment of identification services in a distributed network.

[009] According to an aspect of the invention, a method provides an identification service in a distributed system, comprising providing service elements, each service element including an adapter, a filter, and a logger and receiving, by a first adapter, identification data from a reader. The identification data is provided by the first adapter to a first filter and the first filter processes the identification data. The processed data is provided by the first filter to a first logger, and the first logger notifies

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

a recipient of the processed data. The service elements are monitored to determine whether any service element fails.

[010] In accordance with another aspect of the invention, a method provides an identification service in a distributed system by creating an application corresponding to each of a plurality of service elements, the service elements including an adapter, a filter, and a logger. The adapter application receives identification information corresponding to an item from a reader and provides the identification information to the filter application. The filter application processes the identification information application to create processed information including at least an identification code for the item and provides the processed information to the logger application. The logger application provides the processed information to a recipient. The application corresponding to each service is monitored to determine whether any application fails.

[011] According to another embodiment of the present invention, a system provides a distributed identification service comprising a reader service having service elements comprising an adapter that receives identification information from a reader, a filter that processes the identification information, and a logger that notifies a user of the processed information. The system further comprises a registry service that establishes the reader service and its service elements and a monitor service that determines whether the reader service or any of the service elements fails.

[012] Additional features of the invention will be set forth in part in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

BRIEF DESCRIPTION OF THE DRAWINGS

[013] It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as claimed. The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments of the invention and together with the description, serve to explain the principles of the invention. In the drawings:

[014] Figure 1 is a high level block diagram of an exemplary system for practicing systems and methods consistent with the present invention;

[015] Figure 2 depicts a computer in greater detail to show a number of the software components of an exemplary distributed system consistent with the present invention;

[016] Figure 3 depicts an embodiment of the discovery process in more detail, in accordance with the present invention;

[017] Figure 4 is a flow chart of an embodiment of the event handling process, in accordance with the present invention;

[018] Figure 5 is a block diagram of an exemplary operational string, in accordance with the present invention;

[019] Figure 6 is a block diagram of an exemplary service element, in accordance with the present invention;

[020] Figure 7 depicts a block diagram of a system in which a Jini™ Service Bean (JSB) provides its service to a client, in accordance with the present invention;

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[021] Figure 8 depicts a block diagram of a cybernode in accordance with the present invention;

[022] Figure 9 depicts a block diagram of a system in which a cybernode interacts with a service provisioner, in accordance with the present invention;

[023] Figure 10 is a flow chart of Jini™ Service Bean (JSB) creation performed by a cybernode, in accordance with the present invention;

[024] Figure 11 is a block diagram of a service provisioner in greater detail, in accordance with the present invention;

[025] Figure 12 is a flow chart of dynamic provisioning performed by a service provisioner, in accordance with the present invention;

[026] Figure 13 is a block diagram of a system for providing an identification service, in accordance with the present invention;

[027] Figure 14 is a block diagram of an identification service in greater detail, in accordance with the present invention; and

[028] Figure 15 is a block diagram of a reader service in greater detail, in accordance with the present invention.

DETAILED DESCRIPTION

[029] The following description of embodiments of this invention refers to the accompanying drawings. Where appropriate, the same reference numbers in different drawings refer to the same or similar elements.

C. Introduction

[030] Systems consistent with the present invention provide identification services over a distributed network through a collection of service components that

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

interface with identification devices and process data received from the devices. For example, items may be uniquely identified by an electronic product code (EPC) encoded in RFID tags. A reader device may detect the presence of an RFID tag and scan the EPC and any other information contained in the RFID tag. The data may then be used by applications such as an inventory system or a retail system to track and manage the items. One such identification network is described in the Auto-ID Savant Specification 1.0, available at www.autoidcenter.org.

[031] An identification service consistent with an embodiment of the present invention provides a distributed middleware component between reader devices and users of identification data such as applications, databases, etc. By its nature, an identification service using RFID technology is typically widely distributed. Readers may be separated by a great distance from the users of identification data. For example, an international retailer may need to track inventory in stores around the world using computer systems located in a central headquarters. In another example, a shipping company may use RFID readers throughout a series of processing facilities and transportation hubs to track shipments.

[032] One embodiment of the present invention can be implemented using the Rio™ architecture developed by Sun Microsystems and described in greater detail below. Rio uses tools provided by the Jini™ architecture, such as discovery and event handling, to provision and monitor complex services, such as identification services, in a distributed system. Systems consistent with the present invention provide identification services that may be deployed in a distributed system, including tools to deconstruct a complex identification service into service elements, provision service elements that are

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

needed to make up the complex identification service, and monitor the service elements to ensure that the complex identification service is supported in a dynamic, robust fashion.

B. Exemplary Distributed System

[033] Figure 1 is a high-level block diagram of an exemplary distributed system consistent with the present invention. Figure 1 depicts a distributed system 100 that includes computers 102 and 104 and a device 106 communicating via a network 108. Computers 102 and 104 can use any type of computing platform. Device 106 may be any of a number of devices, such as a reader device, printer, fax machine, storage device, or computer. Network 108 may be, for example, a local area network, wide area network, or the Internet. Although only two computers and one device are depicted in distributed system 100, one skilled in the art will appreciate that distributed system 100 may include additional computers and/or devices.

[034] The computers and devices of distributed system 100 provide services to one another. A "service" is a resource, data, or functionality that can be accessed by a user, program, device, or another service. Typical services include devices, such as reader devices, printers, displays, and disks; software, such as applications or utilities; and information managers, such as databases and file systems. These services may appear programmatically as objects of the Java programming environment and may include other objects, software components written in different programming languages, or hardware devices. As such, a service typically has an interface defining the operations that can be requested of that service.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[035] Figure 2 depicts computers 102 in greater detail to show a number of the software components of distributed system 100. One skilled in the art will recognize that computer 104 and device 106 could be similarly configured. Computer 102 contains a memory 202, a secondary storage device 204, a central processing unit (CPU) 206, an input device 208, and output device 210. Memory 202 includes a look-up service 212, a discovery server 214, and a Java runtime system 216. Java runtime system 216 includes Remote Method Invocation (RMI) process 218 and Java virtual machine (JVM) 220. Secondary storage device 204 includes a Java space 222.

[036] Memory 202 can be, for example, a random access memory. Secondary storage device 204 can be, for example, a CD-ROM. CPU 206 can support any platform compatible with JVM 220. Input device 208 can be, for example, a keyboard or mouse. Output device 210 can be, for example, a printer.

[037] JVM 220 acts like an abstract computing machine, receiving instructions from programs in the form of bytecodes and interpreting these bytecodes by dynamically converting them into a form for execution, such as object code, and executing them. RMI 218 facilitates remote method invocation by allowing objects executing on one computer or device to invoke methods of an object on another computer or device. Lookup Service 212 and Discovery Server 214 are described in detail below. Java space 222 is an object repository used by programs within distributed system 100 to store objects. Programs use Java space 222 to store objects persistently as well as to make them accessible to other devices within distributed system 100.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

C. The Jini Environment

[038] The Jini environment enables users to build and maintain a network of services running on computers and devices. Jini is an architectural framework provided by Sun Microsystems that provides an infrastructure for creating a flexible distributed system. In particular, the Jini architecture enables users to build and maintain a network of services on computers and/or devices. The Jini architecture includes Lookup Service 212 and Discovery Server 214 that enable services on the network to find other services and establish communications directly with those services.

[039] Lookup Service 212 defines the services that are available in distributed system 100. Lookup Service 212 contains one object for each service within the system, and each object contains various methods that facilitate access to the corresponding service. Discovery Server 214 detects when a new device is added to distributed system 100 during a process known as boot and join, or discovery. When a new device is detected, Discovery Server 214 passes a reference to the new device to Lookup Service 212. The new device may then register its services with Lookup Service 212, making the device's services available to others in distributed system 100. One skilled in the art will appreciate that exemplary distributed system 100 may contain many Lookup Services and Discovery Servers.

[040] Figure 3 depicts an embodiment of the discovery process in more detail. This process involves a service provider 302, a service consumer 304, and a lookup service 306. One skilled in the art will recognize that service provider 302, service consumer 304, and lookup service 306 may be objects running on computer 102, computer 104, or device 106.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[041] Service provider 302 discovers and joins lookup service 306, making the services provided by service provider 302 available to other computers and devices in the distributed system. When service consumer 304 requires a service, it discovers lookup service 306 and sends a lookup request specifying the needed service to lookup service 306. In response, lookup service 306 returns a proxy that corresponds to service provider 302 to service consumer 304. The proxy enables service consumer 304 to establish contact directly with service provider 302. Service provider 302 is then able to provide the service-to-service consumer 304 as needed. An implementation of the lookup service is explained in "The Jini™ Lookup Service Specification," contained in Arnold et al., The Jini™ Specification, Addison-Wesley, 1999, pp. 217-231.

[042] Distributed systems that use the Jini architecture often communicate via an event handling process that allows an object running on one Java virtual machine (i.e., an event consumer or event listener) to register interest in an event that occurs in an object running on another Java virtual machine (i.e., an event generator or event producer). An event can be, for example, a change in the state of the event producer. When the event occurs, the event consumer is notified. This notification can be provided by, for example, the event producer.

[043] Figure 4 is a flow chart of one embodiment of the event handling process. An event producer that produces event A registers with a lookup service (step 402). When an event consumer sends a lookup request-specifying event A to the lookup service (step 404), the lookup service returns a proxy for the event producer for event A to the event consumer (step 406). The event consumer uses the proxy to register with the event producer (step 408). Each time the event occurs thereafter, the event

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

producer notifies the event consumer (step 410). An implementation of Jini event handling is explained in "The Jini™ Distributed Event Specification," contained in Arnold et al., The Jini™ Specification, Addison-Wesley, 1999, pp. 155-182.

D. Overview of Rio™ Architecture

[044] The Rio architecture enhances the basic Jini architecture to provision and monitor complex services by considering a complex service as a collection of service elements. To provide the complex service, the Rio architecture instantiates and monitors a service instance corresponding to each service element. A service element might correspond to, for example, an application service or an infrastructure service. In general, an application service is developed to solve a specific application problem, such as processing identification information, word processing, or spreadsheet management. An infrastructure service, such as the Jin lookup service, provides the building blocks on which application services can be used. One implementation of the Jin lookup service is described in U.S. Patent No. 6,185,611, for "Dynamic Lookup Service in a Distributed System," which is incorporated herein by reference.

[045] Consistent with the present invention, a complex service can be represented by an operational string. Figure 5 depicts an exemplary operational string 502 that includes one or more service elements 506 and another operational string B 504. Operational string B 504 in turn includes additional service elements (3, 4, ... n) 506. For example, operational string 502 might represent an identification service. Service element 1 might be an interface with a reader device and service element 2 might be an interface with a user of identification data. Operational string B might be a service to process data received from the reader device. Service element 3 might then

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

be a filter to remove unnecessary data, service element 4 might be a queue storing filtered data, etc. In an embodiment of the present invention, an operational string can be expressed as an XML document. It will be clear to one of skill in the art that an operational string can contain any number of service elements and operational strings.

[046] Figure 6 is a block diagram of a service element in greater detail. A service element contains instructions for creating a corresponding service instance. In one implementation consistent with the present invention, service element 506 includes a service provision management object 602 and a service bean attributes object 604. Service provision management object 602 contains instructions for provisioning and monitoring the service that corresponds to service element 506. For example, if the service is a software application; these instructions may include the requirements of the software application, such as hardware requirements, response time, throughput, etc. Service bean attributes object 604 contains instructions for creating an instance of the service corresponding to service element 506. In one implementation consistent with the present invention, a service instance is referred to as a Jini™ Service Bean (JSB).

E. Jini™ Service Beans

[047] A Jini™ Service Bean (JSB) is a Java object that provides a service in a distributed system. As such, a JSB implements one or more remote methods that together constitute the service provided by the JSB. A JSB is defined by an interface that declares each of the JSB's remote methods using Jini™ Remote Method Invocation (RMI) conventions. In addition to its remote methods, a JSB may include a proxy and a user interface consistent with the Jini architecture.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

[048] Figure 7 depicts a block diagram of a system in which a JSB provides its service to a client. This system includes a JSB 702, a lookup service 704, and a client 706. When JSB 702 is created, it registers with lookup service 704 to make its service available to others in the distributed system. When a client 706 needs the service provided by JSB 702, client 706 sends a lookup request to lookup service 702 and receives in response a proxy 708 corresponding to JSB 706. Consistent with an implementation of the present invention, a proxy is a Java object, and its types (i.e., its interfaces and superclasses) represent its corresponding service. For example, a proxy object for a reader device would implement a reader device interface. Client 706 then uses JSB proxy 708 to communicate directly with JSB 702 via a JSB interface 710. This communication enables client 706 to obtain the service provided by JSB 702. Client 706 may be, for example, a process running on computer 102, and JSB 702 may be, for example, a process running on device 106.

F. Cybernode Processing

[049] A JSB is created and receives fundamental life-cycle support from an infrastructure service called a "cybernode." A cybernode runs on a compute resource, such as a computer or other data processing device. In one embodiment of the present invention, a cybernode runs as a Java virtual machine, such as JVM 220, on a computer, such as computer 102. Consistent with the present invention, a compute resource may run any number of cybernodes at a time and a cybernode may support any number of JSBs.

[050] Figure 8 depicts a block diagram of a cybernode. Cybernode 801 includes service instantiator 802 and service bean instantiator 804. Cybernode 801 may also

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

include one or more JSBs 806 and one or more quality of service (QoS) capabilities 808. QoS capabilities 808 represent the capabilities, such as CPU speed, disk space, connectivity capability, bandwidth, etc., of the compute resource on which cybernode 801 runs.

[051] Service instantiator object 802 is used by cybernode 801 to register its availability to support JSBs and to receive requests to instantiate JSBs. For example, using the Jini event handling process, service instantiator object 802 can register interest in receiving service provision events from a service provisioner, discussed below. A service provision event is typically a request to create a JSB. The registration process might include declaring QoS capabilities 808 to the service provisioner. These capabilities can be used by the service provisioner to determine what compute resource, and therefore what cybernode, should instantiate a particular JSB, as described in greater detail below. In some instances, when a compute resource is initiated, its capabilities are declared to the cybernode 801 running on the compute resource and stored as QoS capabilities 808.

[052] Service bean instantiator object 804 is used by cybernode 801 to create JSBs 806 when service instantiator object 804 receives a service provision event. Using JSB attributes contained in the service provision event, cybernode 801 instantiates the JSB, and ensures that the JSB and its corresponding service remain available over the network. Service bean instantiator object 804 can be used by cybernode 801 to download JSB class files from a code server as needed.

[053] Figure 9 depicts a block diagram of a system in which a cybernode interacts with a service provisioner. This system includes a lookup service 902, a

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

cybernode 801, a service provisioner 906, and a code server 908. As described above, cybernode 801 is an infrastructure service that supports one or more JSBs. Cybernode 801 uses lookup service 902 to make its services (i.e., the instantiation and support of JSBs) available over the distributed system. When a member of the distributed system, such as service provisioner 906, needs to have a JSB created, it discovers cybernode 801 via lookup service 902. In its lookup request, service provisioner 906 may specify a certain capability that the cybernode should have. In response to its lookup request, service provisioner 906 receives a proxy from lookup service 902 that enables direct communication with cybernode 801.

[054] Figure 10 is a flow chart of JSB creation performed by a cybernode. A cybernode, such as cybernode 801, uses lookup service 902 to discover one or more service provisioners 906 on the network (step 1002). Cybernode 801 then registers with service provisioners 906 by declaring the QoS capabilities corresponding to the underlying compute resource of cybernode 801 (step 1004). When cybernode 801 receives a service provision event containing JSB requirements from service provisioner 906 (step 1006), cybernode 801 may download class files corresponding to the JSB requirements from code server 908 (step 1008). Code server 908 may be, for example, an HTTP server. Cybernode 801 then instantiates the JSB (step 1010).

[055] As described above, JSBs and cybernodes comprise the basic tools to provide a service corresponding to a service element in an operational string consistent with the present invention. A service provisioner for managing the operational string itself will now be described.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

G. Dynamic Service Provisioning

[056] A service provisioner is an infrastructure service that provides the capability to deploy and monitor operational strings. As described above, an operational string is a collection of service elements that together constitute a complex service in a distributed system. To manage an operational string, a service provisioner determines whether a service instance corresponding to each service element in the operational string is running on the network. The service provisioner dynamically provisions an instance of any service element not represented on the network. The service provisioner also monitors the service instance corresponding to each service element in the operational string to ensure that the complex service represented by the operational string is provided correctly.

[057] Figure 11 is a block diagram of a service provisioner in greater detail. Service provisioner 906 includes a list 1102 of available cybernodes running in the distributed system. For each available cybernode, the QoS attributes of its underlying compute resource are stored in list 1102. For example, if an available cybernode runs on a computer, then the QoS attributes stored in list 1102 might include the computer's CPU speed or storage capacity. Service provisioner 406 also includes one or more operational strings 1104.

[058] Figure 12 is a flow chart of dynamic provisioning performed by a service provisioner. Service provisioner 906 obtains an operational string consisting of any number of service elements (step 1202). The operational string may be, for example, operational string 502 or 504. Service provisioner 906 may obtain the operational string from, for example, a programmer wishing to establish a new service in a distributed

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

system. For the first service in the operational string, service provisioner 906 uses a lookup service, such as lookup service 902, to discover whether an instance of the first service is running on the network (step 1204). If an instance of the first service is running on the network (step 1206), then service provisioner 906 starts a monitor corresponding to that service element (step 1208). The monitor detects, for example, when a service instance fails. If there are more services in the operational string (step 1210), then the process is repeated for the next service in the operational string.

[059] If an instance of the next service is not running on the network (step 1206), then service provisioner 906 determines a target cybernode that matches the next service (step 1212). The process of matching a service instance to a cybernode is discussed below. Service provisioner 906 fires a service provision event to the target cybernode requesting creation of a JSB to perform the next service (step 1214). In one embodiment, the service provision event includes service bean attributes object 604 from service element 506. Service provisioner 906 then uses a lookup service to ' discover the newly instantiated JSB (step 1216) and starts a monitor corresponding to that JSB (step 1208).

[060] Once a service instance is running, service provisioner 906 monitors it and directs its recovery if the service instance fails for any reason. For example, if a monitor detects that a service instance has failed, service provisioner 906 may issue a new service provision event to create a new JSB to provide the corresponding service. In one embodiment of the present invention, service provisioner 906 can monitor services that are provided by objects other than JSBs. The service provisioner therefore

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

provides the ability to deal with damaged or failed resources while supporting a complex service.

[061] Service provisioner 906 also ensures quality of service by distributing a service provision request to the compute resource best matched to the requirements of the service element. A service, such as a software component, has requirements, such as hardware requirements, response time, throughput, etc. In one embodiment of the present invention, a software component provides a specification of its requirements as part of its configuration. These requirements are embodied in service provision management object 602 of the corresponding service element. A compute resource may be, for example, a computer or a device, with capabilities such as CPU speed, disk space, connectivity capability, bandwidth, etc.

[062] In one implementation consistent with the present invention, the matching of software component to compute resource follows the semantics of the Class.isAssignable() method, a known method in the Java programming language. If the class or interface represented by QoS class object of the software component is either the same as, or is a superclass or superinterface of, the class or interface represented by the class parameter of the QoS class object of the compute resource, then a cybernode resident on the compute resource is invoked to instantiate a JSB for the software component. Consistent with the present invention, additional analysis of the compute resource may be performed before the "match" is complete. For example, further analysis may be conducted to determine the compute resource's capability to process an increased load or adhere to service level agreements required by the software component.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

H. Enhanced Event Handling

[063] Systems consistent with the present invention may expand upon traditional Jini event handling by employing flexible dispatch mechanisms selected by an event producer. When more than one event consumer has registered interest in an event, the event producer can use any policy it chooses for determining the order in which it notifies the event consumers. The notification policy can be, for example, round robin notification, in which the event consumers are notified in the order in which they registered interest in an event, beginning with the first event consumer that registered interest. For the next event notification, the round robin notification will begin with the second event consumer in the list and proceed in the same manner. Alternatively, an event producer could select a random order for notification, or it could reverse the order of notification with each event.

[064] In an embodiment of the present invention, a service provisioner is an event producer and cybernodes register with it as event consumers. When the service provisioner needs to have a JSB instantiated to complete an operational string, the service provisioner fires a service provision event to all of the cybernodes that have registered, using an event notification scheme of its choosing.

I. Identification Service

[065] Figure 13 is a block diagram of one embodiment of the present invention that leverages the capabilities of the Jini and Rio systems to provide an identification service in a distributed network. One or more readers 1302 may detect and scan RFID tags that come within a predetermined range of a reader. The RFID tag may include data, including, for example, an electronic product code (EPC) identifying an item. The

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

RFID data may also include, for example, a location of the reader, a data/time the tag was read, and an identifier of the ready. Reader 1302 may capture this data and transmit it to an identification service 1304, which may process the data and provide it to users 1306. Users 1306 may be, for example, applications or databases that manage and store the processed data.

[066] Readers 1302 may be, for example, any type of device that reads or senses data, such as identification data. For example, readers 1302 may be conventional RFID readers such as the Alien Nanoscanner Reader, the ThingMagic Mercury 3 RFID Reader, or the Matrics Advanced Reader. A large identification system may have hundreds or even thousands of readers, so readers 1302 are typically simple, inexpensive devices that perform only basic data processing. Identification service 1304 may provide extensive data processing to transform the largely untouched scanned data from readers 1302 into useful data customized to meet the preferences and requirements of user 1306. One skilled in the art will appreciate that the system of Figure 13 may include any number of readers 1302, identification services 1304, and users 1306. Consistent with the present invention, readers 1302 may also include devices that sense data, such as motion sensors, proximity sensors, or temperature sensors. Other types of devices may also be used without departing from the spirit and scope of the present invention.

[067] Figure 14 is a block diagram showing identification service 1304 in greater detail. As shown in Figure 13, identification service 1304 may include components such as reader services 1402 and application service 1404. In one embodiment, reader services 1402 may include interfaces to control, manage, and secure devices such as

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

readers 1302. In this embodiment, application service 1404 may include interfaces to customize data for and communicate with user 1306.

[068] Identification service 1304 may also include a monitor service 1406, a registry service 1408, and a code server 1410 to support the provisioning of service components for identification service 1304. In one embodiment of the present invention, monitor service may be a Rio™ monitor implemented, for example, by service provisioner 906. As discussed above, a Rio™ monitor may detect when a service fails to ensure robust service provision. In this embodiment, registry service 1408 may be implemented as a Jini lookup service 212, described above, that enables services to establish communications with other services, and code server 1410 may be implemented as a code server 908, described above, that stores Java code files. Although Figure 14 shows two reader services 1402, one application service 1404, one monitor service 1406, one registry service 1408, and one code server 1410, one skilled in the art will appreciate that identification service 1304 may have any number, including zero, of each of these components.

[069] Using the Rio architecture, identification service 1304 may be implemented using an operational string as described above. Service elements of the operational string may be reader service 1042 and application service 1404. In this way, monitor service 1406, registry service 1408, and code server 1410 may be used to provision and monitor identification service 1034 to ensure a robust and flexible distributed implementation.

[070] Figure 15 is a block diagram of service such as reader service 1402 in greater detail. Reader service 1402 may include components such as adapter 1502,

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

filter 1504, and logger 1506. Adapter 1502 may interface with an outside device such as reader 1302 to receive RFID data. Filter 1504 may process the data received from adapter 1502, and logger 1506 may communicate the processed data, for example, to user 1306. One skilled in the art will appreciate that these components may be linked together in the order shown or in any other order. In an alternative embodiment, reader service 1402 may include a queue 1508 (not pictured) as described below. Each of these components may be expressed using, for example, an XML document.

[071] Application service 1404 may be implemented using similar components to those described in reader service 1402. Application service may include, for example, adapter 1502, filter 1504, logger 1506, and/or queue 1508 (not shown) to facilitate interaction with user 1306. As discussed above, user 1306 may be, for example, an application or a database.

[072] Adapter 1502 may be, for example, a device adapter to interface with any type of reader device. This increases the feasibility and economy of identification service 1304. In an embodiment of the present invention, adapter 1502 may link to other components, such as filter 1504, logger 1506, or queue 1508 (not shown), as an event producer using the enhanced event handling described above. An adapter may include a unique name, a Java class that implements the adapter, a set of properties, and a set of outputs. The properties may be, for example, a sequence of name value pairs that are passed to the adapter's constructor. The outputs may be, for example, a sequence of component names to be registered as event listeners to the adapter. The outputs may designate, for example, filter 1504, logger 1506, or queue 1508 (not

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

shown). The following is program code for an exemplary configuration of adapter 1502 written as an XML document:

```
<ems:adapter>
        <ems:name>TestReader</ems:name>
        <ems:classname>com.sun.autoid.adapter.TestAdapter</ems:classmate>
        <ems:properties>
                <ems:property>LogLevel</ems:property>
                <ems:value>INFO</ems:value>
        </ems:properties>
        <ems:properties>
                <ems:property>host</ems:property>
                <ems:value>localhost</ems:value>
        </ems:properties>
        <ems:properties>
                <ems:property>port</ems:property>
                <ems:value>5050</ems:value>
        </ems:properties>
        <ems:outputs>
                <ems:output>Smoother</ems:output>
        </ems:outputs>
</ems:adapter>
```

[073] In an embodiment of the invention, filter 1504 may be used to process data received from reader 1302. RFID tags generate a much larger amount of data than traditional bar codes. Much of this data is often considered to be "noise," meaning that it is of little or no use. Filter 1504 is designed to separate useful data from noise. Filter 1504 may also detect and remove data from mis-read RFID tags, increasing the accuracy of the resulting data. Filter 1504 may link to other components as an event listener, using the enhanced event handling described above. A filter may include a unique name, a Java™ class that implements the filter, a set of properties, and a set of outputs. The properties may be, for example, a sequence of name value pairs that are passed to the filter's constructor. The outputs may be, for example, a sequence of component names that will be registered as event listeners to the filter. The outputs

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

may designate, for example, other filters 1504, logger 1506, or queue 1508 (not shown). The following is program code for an exemplary configuration of filter 1504 written as an XML document:

```
<ems:filter>
        <ems:name>Smoother</ems:name>
        <ems:classname>com.sun.autoid.filter.Smoothing</ems:classname>
        <ems:properties>
                <ems:property>LogLevel</ems:property>
                <ems:value>INFO</ems:value>
        </ems:properties>
        <ems:properties>
                <ems:property>MaxCycles</ems:property>
                <ems:value>10</ems:value>
</ems:properties>
<ems:outputs>
                <ems:output>EventQueue</ems:output>
        </ems:outputs>
</ems:filter>
```

[074] A queue 1508 (not shown) may be implemented as both an event listener and an event producer. In this way, for example, queue 1508 may receive events from adapter 1502 and send events to filter 1504 or logger 1506. Queue 1508 may include a unique name, a Java™ class that implements the filter, a set of properties, and a set of outputs. The properties may be, for example, a sequence of name value pairs that are passed to the queue's constructor. The outputs may be, for example, a sequence of component names that will be registered as event listeners and/or event producers to the queue. The outputs may designate, for example, adapters 1502, filters 1504, or loggers 1506. The following is program code for an exemplary configuration of filter 1508 written as an XML document:

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

```
<ems:queue>
        <ems:name>EventQueue </ems:name>
       <ems:classname>com.sun.autoid.queue.MultiThreadedQueue</ems:classname>
        <ems:properties>
                <ems:property>LogLevel</ems:property>
                <ems:value>INFO</ems:value>
</ems:properties>
<ems:properties>
                <ems:property>MaxThreads</ems:property>
                <ems:value>10</ems:value>
        </ems:properties>
        <ems:outputs>
                <ems:output>EventLogger </ems:output>
                <ems:output>FileLogger</ems:/output>
        </ems:outputs>
</ems:queue>
```

[075] In an embodiment of the present invention, logger 1506 may be implemented to notify user 1306 of data from readers 1302. Logger 1506 may notify user 1306 of RFID and non-RFID events using a protocol specified by user 1306, for example, by logging information to a file system, a JMS queue, or a XML/HTTP file. Logger 1506 may function as an event listener (e.g., for data from filters) and/or an event producer (e.g., for data for users). Logger 1506 may include, for example, a unique name, a Java™ class that implements the logger, and a set of properties. The properties may be, for example, a sequence of name value pairs that are passed to the logger's constructor. The following is program code for an exemplary configuration of filter 1506 written as an XML document:

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

```
<ems:logger>
        <ems:name>EventLogger</ems:name>
        <ems:classname>com.sun.autoid.logger.REProduce</ems:classname>
        <ems:properties>
                <ems:property>LogLevel</ems:property>
                <ems:value>ALL</ems:value>
        </ems:properties>
        <ems:properties>
                <ems:property>EventID</ems:property>
                <ems:value>97531</ems:value>
        </ems:properties>
</ems:logger>
```

[076] In a distributed system, an identification service may be implemented using the Rio architecture described above to provide robust and efficient processing of data from a reader device for a user. Additional services may also be provided, such as an enterprise gateway including service interfaces to enable applications to register and receive identification data in useful ways. For example, a user of the enterprise gateway may request notification of the number of items at a particular location, the location of a specific item, or the path an item has taken. The gateway may be implemented, for example, as a web interface.

[077] The foregoing description of an implementation of the invention has been presented for purposes of illustration and description. It is not exhaustive and does not limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practicing of the invention. Additional modifications and variations of the invention may be, for example, the described implementation includes software but the present invention may be implemented as a combination of hardware and software or in hardware alone. The

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com

invention may be implemented with both object-oriented and non-object-oriented programming systems.

[078] For example, identification services consistent with the present invention be implemented using, for example, the Java Management Extensions (JMX) technology to provide identification services that are accessible by management and monitoring applications. JMX Managed Beans may be created to implement components of the identification service and registered with a Jini lookup service. In this way, the JMX Managed Beans could be discovered and accessed by management applications. The JMX technology is described further at http://java.sun.com/products/JavaManagement/. The JMX technology may be implemented, for example, using the Java Dynamic Management Kit (JDMK).

[079] Identification services consistent with embodiments of the present invention may also be implemented using Simple Network Management Protocol (SNMP) Management Beans that may be accessible to network management systems such as Openview, Tivoli, and BMC Patrol. In this way, an identification service may be provided with unique management and monitoring capabilities.

[080] Furthermore, one skilled in the art would recognize the ability to implement the present invention in many different situations. For example, the present invention can be applied to the telecommunications industry. A complex service, such as a telecommunications customer support system, may be represented as a collection of service elements such as customer service phone lines, routers to route calls to the appropriate customer service entity, and billing for customer services provided. The present invention could also be applied in any event processing system.

FINNEGAN
HENDERSON
FARABOW
GARRETT &
DUNNER LLP

1300 I Street, NW
Washington, DC 20005
202.408.4000
Fax 202.408.4400
www.finnegan.com